03-10-00

A

KAPLAN & GILMAN, L.L.P.

COUNSELORS AT LAW

MICHAEL R. GILMAN†
JEFFREY I KAPLAN

OF COUNSEL
RONALD B GOLDSTEIN
FRANCINE M. MEYER

† ADMITTED ONLY IN NY & CT

900 ROUTE 9 NORTH
WOODBRIDGE, NEW JERSEY 07095
TELEPHONE (732) 634-7634
FACSIMILE (732) 634-6887

73 CROTON AVENUE
OSSINING, NEW YORK 10562
TELEPHONE (914) 923-6240
FACSIMILE (914) 923-6258

Date: March 9, 2000

Re: Inventor(s): Sam Mazza

Title: **DATA CONVERSION**

Atty. Docket No.: 024/1

Box NEW APP. FEE
Assistant Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

Submitted herewith is the above-identified patent application. Also enclosed are:

1) A self-addressed, stamped return postcard; and

2) 1 sheet of an informal drawing of Fig. 1.

Respectfully submitted,

KAPLAN & GILMAN, L.L.P.

Jeffrey I. Kaplan
Reg. No. 34,356

JIK/pa
Enclosures

CERTIFICATE OF MAILING

Express Mail mailing label number: EL390696055US
Date of Deposit: March 9, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service *Express Mail Post Office to Addressee* service under 37 C.F.R §1.10 on the date indicated above and is addressed to Box NEW APP. FEE, Assistant Commissioner for Patents, Washington, D.C. 20231.

Paula M. Halsey
(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

DATA CONVERSION

TECHNICAL FIELD

This invention relates to data collection and interpretation, and more particularly, to a method of converting unformatted data into a format that is viewable to a viewing entity or a variety of such entities.

BACKGROUND OF THE INVENTION

In voice processing and similar systems, often system data must be formatted and viewed by operators.

When an application presents data to a viewing entity (e.g., a printer), the data typically is outputted in a format that can be understood by the viewing entity. The application may output the formatted data directly to the viewing entity, or store the formatted data in an external storage for later inspection or presentation by the viewing entity. It is expected that, with the help of a viewer (e.g., a printer software), this viewing entity is capable of deciphering the data format as outputted for processing or display. In addition, data conversion is also required for the sharing of data stored among different writing entities.

Data conversion or data formatting, which is typically carried out by the writing entity since it is in possession of the client code of the application, is an expensive operation. In addition, the formatted data also requires more storage than the unformatted data. These costs are further increased in a situation where the same data or portions thereof need to be converted into various formats required by different viewers. For example, when a running program outputs data at various terminals, the same data or some of the data portions are required to be printed or presented at different printers or screens, which may be using different viewers.

Furthermore, when a running program is outputting formatted data, the system performance may be adversely affected by the data formatting process. For example, when a program is running for debugging purpose, events are outputted at various portions of the application by printing out or presenting the formatted data portions at different printers or screens for further analysis. However, the formatting affects the "real time" operation of the system. Hence, a software bug may change behavior due to the printing of the data needed to diagnose the problem.

It has been proposed that the data be formatted by the viewer at the viewing entity, so as to overcome the discussed above problems because the data would be outputted in its unformatted state. However, the viewer cannot by itself understand the objects (data) in the log output by a typical voice processing application. It must use the help of reader servants customized for each type of objects. The disadvantages are in that the viewer would be complex.

SUMMARY OF THE INVENTION

The present invention is directed to the above problems existing in the prior art.

According to the invention, the formatting of the data is not carried out at the writing entity side, but is the responsibility of the viewing entity. The data, which includes a plurality of unformatted data portions, is provided to the viewing entity in an unformatted style. A plurality of formatters, each of which specializes in formatting one or more of the data portions into a format viewable to a viewer at the viewing entity, are located for each of the data portions at the viewing entity. The data portions are thereafter converted into the format by said formatters and presented to the viewer.

Identifying means, such as tags, is provided to identify each of the data formatted. The

formatters for each data portion may be located by its identifying tag. In a preferred embodiment, the locating of the formatters is carried out by the viewer and the formatters are plug-able into the viewer.

In this way, no data conversion is required at the writing entity since data is outputted simply in its unformatted style. No client code is required to prepare the object for output and therefore any object type may be written with a simple interface. Less storage space is required for storing the data that is now stored in its unformatted style. Data portions of different formats from different writing entities may share the same storage because the conversion of these unformatted data portions is carried out by their respective formatters at the viewing entity.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure one is a conceptual block diagram of the functional components of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Generally, when information or data is presented to a viewing entity (printer, screen, etc), the data should be presented in a human readable form, i.e., in a format viewable to the viewing entity with the help of a viewer (viewing software). Usually the conversion of the data is carried out by the writing entity before it is outputted from a program, with the client code included in the object for output.

In the present invention, the data, which may comprise a plurality of unformatted data portions, is output in its unformatted style to the viewing entity, or may be stored in its unformatted style before being presented to the viewing entity. The unformatted data portions are eventually

converted at the viewing entity into the format required by the viewer of the viewing entity.

These data portions may be in different formats as, for example, they are produced by different writing entities. A plurality of formatters, each of which is capable of formatting one or more of the data portions into the format viewable to the viewer, are also provided to the viewing entity.

Tags are included in the outputted unformatted data for the purpose of identifying each of the data portions. For each of the data portions, a specific formatter, which is specialized in reading this unformatted data portion and conversing it into a format viewable to the viewer, may be located by using the identifying tag of that data portion.

Preferably, the locating of the formatters for the data portions is carried out by the viewer. The formatters are plug-able to the viewer when they are located. These formatters are provided from the writing entity to the viewing entity together with the unformatted data, or may be provided in other way, for example, over Internet. When a formatter is located by the viewer using its identifying tag, the viewer allows the formatter plug in to convert the data portion into the format required by the viewer. If no formatter for a particular data portion can be located by the viewer, this data portion can be skipped with no adverse effect to the performance of the viewer.

With the plug-able formatters specialized in converting different data portions, the viewer does not have to be a complex one. Furthermore, different writing entities can share the data stored..

Figure 1 depicts a conceptual diagram of a viewer 101. The viewer 101 may be implemented in software and resides on a viewing entity such as a thin client, PC computer, etc. In operation, packets or blocks of data are to be processed by entering interpreter 111. The blocks may be in machine readable format. The blocks contain a tag which indicates the type of formatter necessary

to format and interpret the data coming from an arbitrary data source, such as, for example, a hard disk or other bulk storage device.

When it is desired to read in and interpret the data, a detector 102 reads the tag in the header of the first block of data or packet. The tag identifies a type of interpretation and formatting required. The tag may specify a language, whether color or black and white formatting is required, etc. Moreover, the tag may specify that the data is to be interpreted and formatted in several different ways. For example, the interpreter may specify that the incoming data stream is to be interpreted and output in two different formats, one for printing and another for viewing. Alternatively, the different formatters could relate to different languages, print type, etc.

Preferably, a standard interface 107 exists between the formatters 107 and the detector 102. When the detector identifies the one or more formatters requested in the incoming data stream, the appropriate formatters are loaded into interpreter 114. If plural formatters are used, then plural interpreters 114 may exist and receive duplicate copies of the incoming data stream 111 in parallel. In its simplest form, the interpreter may simply be the hardware that runs the formatting software. The formatters are then removed from the interpreter after the data is processed. It is noted that in multitasking systems, there may be times when blocks or packets of data arriving for display are unrelated to each other but instead relate to different tasks being performed. For example, the data stream may contain in sequence, two blocks of data related to a first software application, four more blocks of data related to a second application, then one block related to the first application, and then two more related to the second application. By reading the header in each block, the interpreter can seamlessly interpret and format all of the intermixed blocks of data. The intermixed blocks can be processed by having two interpreters 114. As the detector detects the tag in the header, it routes it to

the proper interpreter.

Alternatively, the a single interpreter can be utilized, with the formatters 103-106 being loaded and reloaded every time the required formatter for incoming packets changes. This latter method is less preferred because it involves the timesharing of the interpreter by two or more different formatters, and the constant replacing in the interpreter 114 of the operative software. This may slow down system performance.

By standardizing the interface 107, various formatters can be written to work with the same interpreter. This means that as new formats are desired, the remainder of the software for processing, interpreting, and displaying the unformatted data need not be rewritten.

In the present invention, the formatters and the viewer are separate from each other until the formatters plug into the viewer. Therefore, it is possible for a data portion to be converted into different formats required by different viewers. In other words, the different formatters may be located by different viewers for the same data portion so as to convert it into different formats that are viewable to respective viewers. It is not necessary for the same data portion to be converted into different formats at the writing entity level, which is an expensive operation.

Because the conversion of the data portions are implemented at the viewing entity, a running program can output unformatted data and no formatting process is necessary. Therefore, there is no adverse timing effects from the formatting process to the system performance.

From the foregoing, it will be observed that numerous variations and modifications may be effected without departing from the true spirit and scope of the novel concept of the invention. For example, the tags can be provided to the viewing entity separately from the data. The formatters may work separately to the viewer instead of a plug-in mode. The viewer may be a generic viewer

available to the public or be one provided by the writing entity specifically for viewing their data. It is intended to cover by the appended claims all such modifications as fall within the scope of the claims.

WHAT IS CLAIMED:

1. A method of presenting data to a viewing entity having a viewer, comprising the steps of:

providing unformatted data to said viewing entity, said data comprising one or more unformatted data portions required to be converted into a format viewable to said viewer;

providing a plurality of formatters, each of which being capable of formatting one or more of said data portions into said format;

locating said formatters by said viewer for each of said unformatted data portions; and

formatting each of said portions by said located formatters whereby said data portions are converted to said format viewable to said viewer.

2. The method of claim 1 further comprising a step or providing identifiers for each of said data portions.

3. The method of claim 2 wherein said identifiers are tags included in said data portions.

4. The method of claim 2 wherein said step of locating formatters is implemented by making use of said identifiers.

5. The method of claim 1 wherein said formatters are plug-able into said viewer.

6. A method of presenting data to a plurality of different viewers, comprising the steps of:

providing unformatted data to each viewers, said data including a plurality of unformatted

data portions;

providing a plurality of formatters, each of which being capable of formatting one or more data portions into at least one format viewable to at least one of said viewers;

locating by each viewer, for each data portion that required to be viewable to said viewer, a formatter capable of conversing said each data portion to a format viewable to said viewer;

formatting said each data portion by said located formatter whereby all of said unformatted data portions can be formatted at relevant viewers by relevant formatters into formats viewable to relevant viewers.

7. A method of claim 6 further comprising a step of providing a plurality of identifiers each of which identifies one of said data portions.

8. A method of claim 7 wherein said identifiers are tags included in relevant data portions.

9. A method of claim 7 wherein said step of locating is implemented by making use of said identifiers of said data portions.

10. A method of claim 6 wherein said formatters are plug-able into each of said viewers that locates them.

11. A system for formatting unformatted data having one or more unformatted portions to be viewable to a viewer, comprising:

conversion means for converting said data portions into a format viewable to said viewer, said conversion means being separately located from said viewer;

identifying means for identifying each of said data portions;

locating means for said viewer, by using said identifying means, to locate said conversion means for each of said data portions whereby said each data portion is converted at said viewer by said conversion means into said format viewable to said viewer.

12. A system of claim 11 wherein said conversion means comprises a plurality of formatters, each of which being capable of converting at least one of said data portions into said format.

13. A system of claim 12 wherein said formatters are plug-able in said viewer.

14. A system of claim 13 wherein said identifying comprises a plurality of tags each of which identifies one of said data portions.

15. A system for interpreting data from machine readable form to at least one human readable format, the system comprising:

a detector for detecting from information in incoming blocks of data one or more formats in which it is desired to display said blocks;

a plurality of formatters, each of the formatters interfacing with the detector using a standard interface;

means within the detector for invoking all formatters required to format said incoming blocks into said one or more formats, and

means for loading said formats into an interpreter, and for subsequently sending said incoming blocks to said interpreter.

16. The system of claim 15 further wherein said means for invoking invokes plural formatters to operate on the same incoming data stream.

17. The system of claim 16 wherein said plural formatters are arranged to receive incoming data in parallel.

18. A system for processing incoming blocks of data wherein the intermixed blocks include blocks to be formatted by different formatters, the system comprising;

a detector for checking a tag contained within each block of data, the tag being indicative of how to format the incoming data;

means responsive to said detector for routing the data to be formatted to proper formatting software; and

means for invoking the proper formatting software using a standard interface common to all of said different formatters to format said incoming blocks of data.

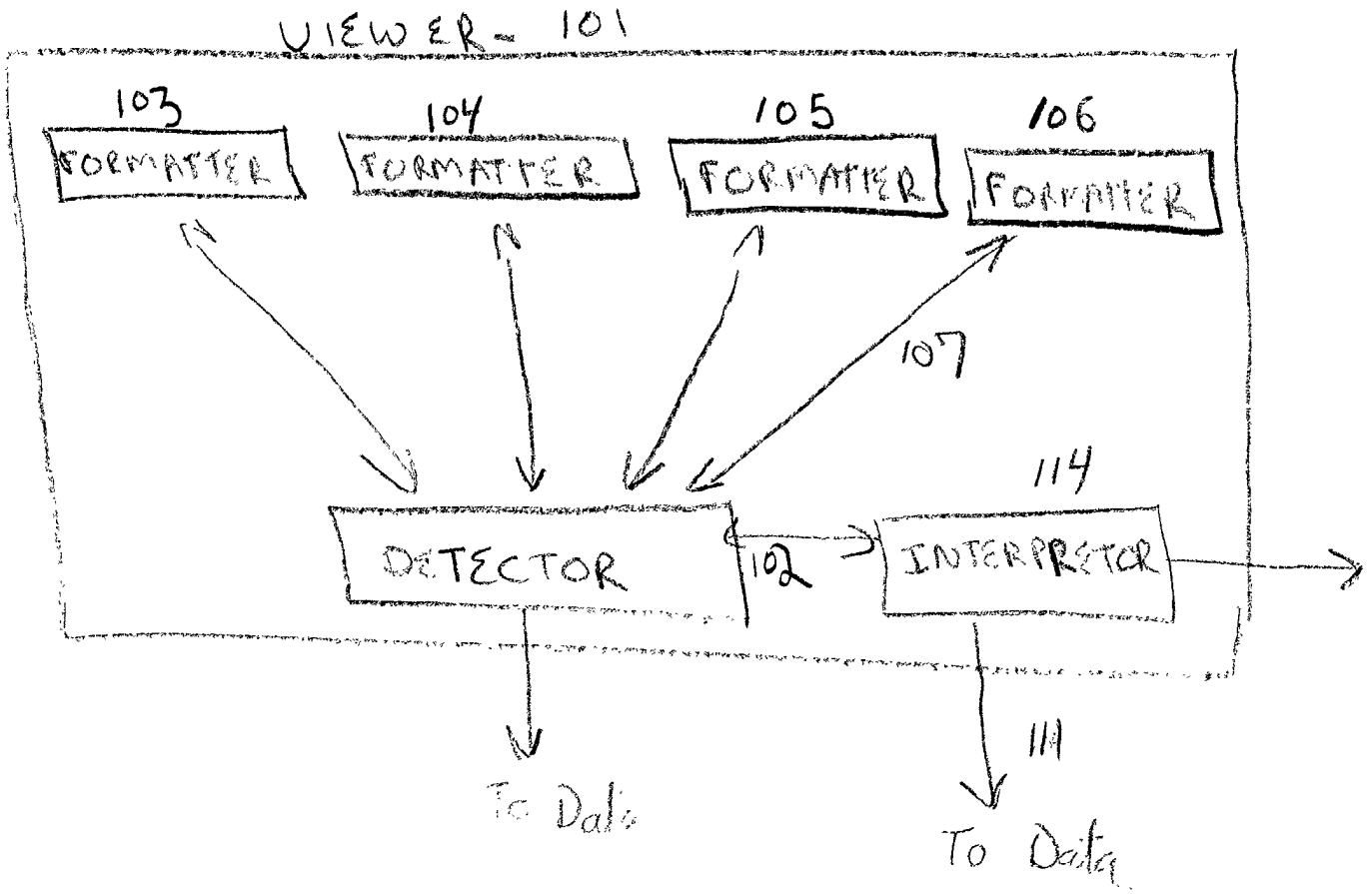
19. The system of claim 18 wherein the means for invoking includes loading software from storage to an interpreter.

20. The system of claim 18 wherein said means for invoking includes a switch for routing incoming data to one of plural preloaded formatters.

ABSTRACT

A method of conversing data to a format viewable to a viewer is disclosed. The unformatted data comprising a plurality of unformatted data trunks is outputted without formatting. The formatting of the data portions is carried out at the viewing entity side by means of a plurality of the formatters, each of which is specialized in formatting one of more of the data trunks. The formatters are located by the viewer with the help of the tags identifying each of the data portions and are plug-able in the viewer. No formatting to the data is required for outputting or storing the data.

FIG. 1


VIEWER - 101
103
FORMATTER
104
FORMATTER
105
FORMATTER
106
FORMATTER
107
DETECTOR
102
114
INTERPRETER
To Data
111
To Data